Comparison of Computer Vision algorithm and model with respect to Robotics research and applications

Yazhuo Cao (UID: 10329221)

1 Introduction

Machine learning methods have been employed for image classification for many years, and they have undoubtedly demonstrated state of the art accuracy, however classical computer vision algorithm still has a place in the state of the art in robotics. To evaluate the performance of the classical computer vision algorithm and deep learning model, one representative algorithm and model will be taken from each category and analysed in this paper focusing on object recognition, and their benefit and issue will be discussed concerning robotic research.

2 Bag of Visual Words

Bag of visual words (BoVW) classification method is a commonly used technique in object recognition. It creates a vocabulary for describing the image in terms of extrapolating qualities.

There are several steps to perform the BoVW classification:

1. Extract local features from the image
2. Quantize the feature space using clustering algorithms to build visual dictionaries
3. Each image is now represented by visual words
4. Extract local features and compare these features with words in the visual dictionaries to create histograms
5. Predict the class of test images compared with each histogram of train images.

2.1 Local Features

The ideal local feature for the feature detector should have several properties: independent from geometric transformations, can be repeated under different viewing conditions and the detection of these features must be accurate and efficient[1].

2.1.1 Feature detection

There are several methods for feature detection, each has a different algorithm and would bring a different result. For example, the Harris corner detector is more likely to detect corners and the Difference-of-Gaussian detector detects blob-like features.

In this paper, the advantage and disadvantages of Scale-Invariant Feature Transform (SIFT) and Harris Corner Detector will be discussed.

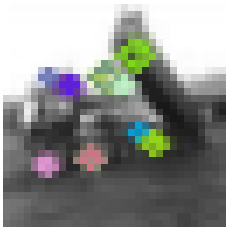
Harris Corner Detector Corners in a grey image are the regions where there is a high change in intensity, they are key points in an image and are invariant to translation, rotation and illumination as required [1]. A mathematical way of finding these corners was purposed in 1988 [2], essentially finding the difference in intensity for a displacement of (u,v) in all directions. The grayscale of the image is taken first, a Gaussian filter is then applied to smooth out any noise and the Sobel operator is used to find the image derivative in both x and y directions for every pixel in the grayscale image, then a window is slid across to find the Harris value R using second-moment matrix. When $R < 0$, the region is flat; if R is small the region is at the edge of a shape; if R is large, it is a corner.

The result of this is a greyscale image with an R score for each pixel. This gave a degree of freedom and responsibility to the user as a threshold is needed to filter the R score. The accuracy of this detection can also be improved using the centroids of the corners to refine them.

Harris Corner Detector requires comparatively little computational resources, therefore it has the advantage of speed when detecting features. However, corners may no longer be the same when the image is scaled therefore the Harris Corner Detector is not scale-invariant. Once detected, the features also need to be passed to a feature descriptor which is more complicated for the user.

Scale-Invariant Feature Transform(SIFT) As the name suggests, SIFT is scale-invariant. Purposed by DG Lowe in 2004 [3], this feature detection method has also proven to be invariant to rotation, illumination, viewpoint, distortion and noise, making it the most popular feature detection method.

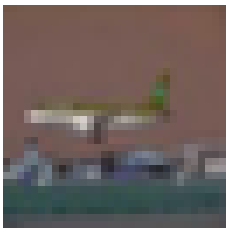
Similar to the Harris Corner Detector, it uses



(a) An image from the CIFAR10 training dataset with features extracted using SIFT Detector labeled.

(b) Same image as 1a with features extracted using Harris Corner Detector labeled.

Figure 1: Two images from the CIFAR10 training dataset with features extracted and labeled



(a) A different image from the CIFAR10 training dataset with no features extracted using SIFT Detector.

(b) Same image as 2a with features extracted using Harris Corner Detector labeled.

Figure 2: Two images from the CIFAR10 training dataset with features extracted and labeled

windows. Scale-space filtering is applied and inside, the Difference of Gaussian(DoG) is used as a blob detector. The Difference of Gaussian is an approximation of Laplacian of Gaussian, the latter is computationally more expensive. The image is then searched for potential key points, known as local extrema over scale and space. They will then be refined using an upper and lower threshold to eliminate low contrast key points and edge key points, leaving only the representative features.

However, the two threshold means some images would have no features extracted, as seen in Figure 2a, where SIFT extracted no features from the image as the contrast between the plane and background is low, whereas Harris Corner Detector extracted several features, as shown in Figure 2b.

2.1.2 Feature representation

SIFT descriptor has been used in the experiments conducted for this paper.

Firstly the scale and orientation of the keypoints are calculated. The descriptor for each keypoint needs to be distinctive and as invariant to changes as possible. A window of 16×16 is taken around the keypoint, split into 16 sub-blocks of 4×4 and an 8 bin orientation histogram is created for each sub-block. This gave a 128 ($4 \times 4 \times 8$)bin value, representing a keypoint descriptor in vector form. This introduces two new dependence of rotation, as orientation is included in the descriptor, and illumination, which needs requires a large threshold number to be independent.

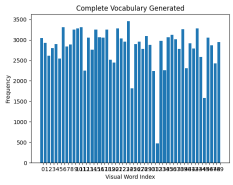
The complicated mathematics involved allow SIFT descriptor to have accurate results, however, due to the computational power needed, SIFT algorithm is slow and ineffective in the context of robotic application where the device power might be low[4].

Another feature representation method is Oriented FAST and Rotated BRIEF(ORB). This is faster than SIFT and is free to use whereas SIFT is patented. ORB uses image pyramid and intensity centroid for scale and rotation invariant, however, is not as robust as SIFT[5].

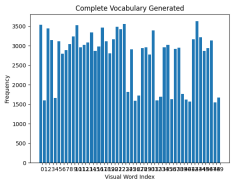
In this paper, features extracted from Harris Corner Detector and SIFT are both passed into SIFT descriptor. This creates a different set of descriptors as shown in Figure 1 while keeping the accuracy of SIFT descriptors with the faster Harris Corners Detector [6]. Features detected by Harris Corners Detector do have the problem of scale-dependent, however, as experiments were done by Pedram Azad(2009) have shown, a scale of 0.75 is the most suitable scale factor when omitting the scale-space analysis to close the gap between adjacent spatial scales.

2.2 Visual dictionaries and image representation

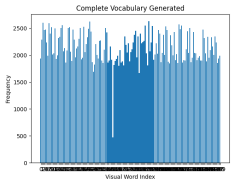
Features need to be clustered into visual/prototype words to reduce computational requirements. Local features will unlikely be the same due to viewpoint, illumination, rotation etc, however, features from the same object would be similar. Therefore if the features are similar, one can assume the local contents are tantamount.



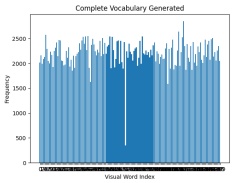
(a) Histogram of 50 visual words using SIFT detector



(b) Histogram of 50 visual words base on Harris Corners Detector



(c) Histogram of 150 visual words using SIFT detector



(d) Histogram of 150 visual words base on Harris Corners Detector

Figure 3: Histogram of visual word vocabulary quantized using K-Mean algorithm

Different clustering algorithms and especially different numbers of words would affect the accuracy of the object recognition.

2.2.1 K-means Clustering

K-means clustering is a simple unsupervised method with certainty in finding clustering centres which in this case are the visual words. Difference cluster widths per dimension can be set to improve the location of the visual words.

However, the number of clustering points needs to be chosen manually by the user and the iterations needed to find the optimal centres are depended on the initial values. In the context of robotics and computer vision, the feature descriptor is a 128-dimension vector. K-mean becomes inefficient at distinguishing between high dimension data due to the curse of dimensionality[7]. To overcome this, a pre-clustering step was needed to reduce the dimensionality of data using PCA(Principal component analysis) and project data into lower dimensional subspace[8].

2.2.2 Histogram Representation

Histogram equalization is a method to process images based on the intensity distribution[9], which in this case, relates to the visual vocabulary. All training and testing images are represented by a histogram of visual words for classification in the next stage to simplify the classification process by only comparing the frequently appeared visual words rather than every feature. This can speed up the classification process and save computational power, crucial in low powered devices such as small size robots, however, the accuracy is heavily dependent on the quality of the visual vocabulary.

The histograms shown in Figure 3 presents the overall frequency of the visual words that appeared in the training set for both SIFT and Harris detection with 50 and 150 visual words. We can see the frequency is uneven, especially shown in Figure 3a at visual word 32. This would have the same effect as an imbalanced training dataset, putting images with visual word 32 at a disadvantage when classifying.

2.3 Category models and classifiers

The category representation needs to be robust to intra-category variation, deformation, and articulation. Two different classifiers, as explained below, were used for this experiment with a range of visual words from 50 to 150, the results are similar between classifiers. As seen in Figure 4, the blue line used the features detected with SIFT with SVM as the classifier, red is Harris with SVM as the classifier, whereas the yellow line used features from the Harris Corners Detector and KNN to predict the result, where k is 1.

2.3.1 Support Vector Machines

When using the Support Vector Machine(SVM) for classification, the model is trained with the visual words from the training data. They are plotted and hyper-planes separates them to the 10 classes. SVM works best when there is a clear separation of classes and is effective in high dimensions space. However, it requires a long training time and is not the best method when target classes overlap, such as this.

2.3.2 K-Nearest Neighbors

K-Nearest Neighbors(KNN) has the advantage of process time since it requires no training, however, this is paid for by the classification step where the distance between a new data and every existing visual words need to be compared, taking $O(N)$ in

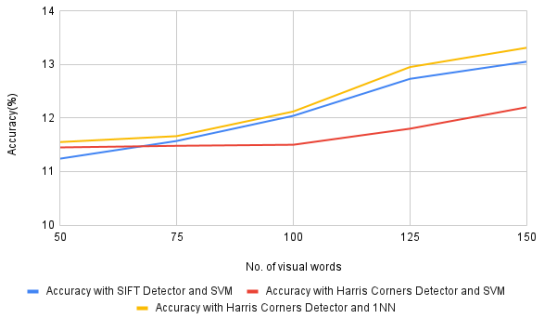


Figure 4: Accuracy of classification using SVM and KNN using SIFT and Harris detector with varies number of visual words

time complexity. This method is also more suitable for lower dimensional spaces because in high dimensions all data look alike[10]. Similar prepossessing methods as mentioned in section 2.2.1 are needed.

2.4 BOVW method used

As mentioned above, Harris Corners Detector and SIFT were used for feature detection, SIFT descriptor was employed for feature representation, K-mean was used for clustering with different numbers of clustering centres considered and the classification used both SVM and KNN.

The limitations of each method were discussed in the corresponding sections above and the results are discussed in 4.2.

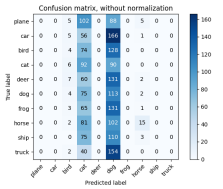
3 Convolutional Neural Network

3.1 Initial model

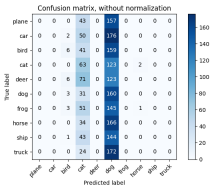
Convolutional Neural Network(CNN) is a class of neural networks that specializes in processing data that has a grid-like topology, which in this case, is the images and their pixel grid.

The RGB values for each pixel of the images are in the range [0, 255] which is not ideal for a neural network due to its size. They were normalized in the range [0, 1] as part of the data preparation before putting them into the CNN model.

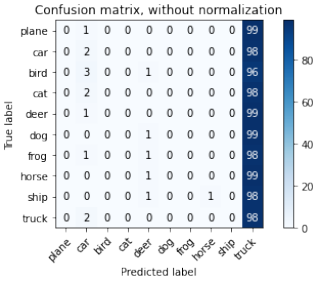
The experiment for the CNN starts with model 1, setting up the input as 32 convolutional filters of size 3×3. The activation function used is ReLU and the network then has a max-pooling layer with



(a) Confusion matrix where SIFT Detector and SVM was use

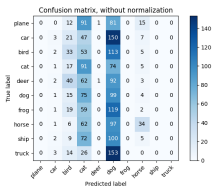


(b) Confusion matrix where Harris Corners Detector and SVM was used

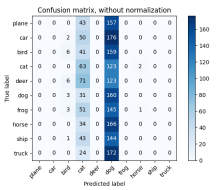


(c) Confusion matrix where SIFT Detector and KNN where K=1

Figure 5: Confusion matrix where 50 visual words were used

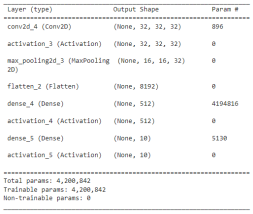


(a) Confusion matrix where SIFT Detector and SVM was use



(b) Confusion matrix where Harris Corners Detector and SVM was used

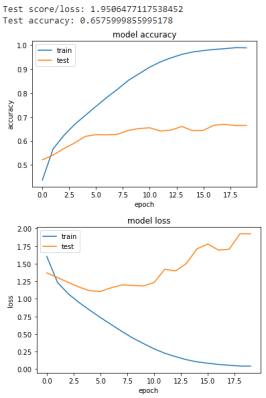
Figure 6: Confusion matrix where 150 visual words were used



Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 32, 32, 32)	896
activation_3 (Activation)	(None, 32, 32, 32)	0
max_pooling2d_3 (MaxPooling 2D)	(None, 16, 16, 32)	0
flatten_2 (Flatten)	(None, 8192)	0
dense_4 (Dense)	(None, 512)	4194816
activation_4 (Activation)	(None, 512)	0
dense_5 (Dense)	(None, 10)	5130
activation_5 (Activation)	(None, 10)	0

Total params: 4,200,842
Trainable params: 4,200,842
Non-trainable params: 0

(a) Model 1 summary



(b) Result after 20 epoch of model shown in Figure 7a

Figure 7: Model 1 and result

pool size of 2×2. The model summary and result are shown in Figure 7a and 7b.

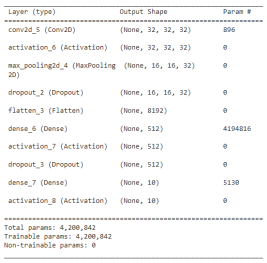
3.2 Dropout layers

As shown in Figure 7b, the testing accuracy does down while the training accuracy goes up. This is an indication of overfitting, likely to happen when the training dataset is small, which is within expectation for reasons mentioned in section 4.1. This can be improved by using the regularization method Dropout to randomly removes nodes from the network temporally [11], which can break up where network layers co-adapt to correct mistakes from the priors layer and simulate a sparse activation. Both make the model more robust. The Dropout Layers are added as shown in Figure 8a and as the accuracy rate on the top of Figure 8b for model 2 has shown, the overall accuracy of the model for testing data has not improved but the accuracy of the training data decreased. This is sign that the addition of Dropout layers stopped the model from overfitting to the training data and makes it more generalised to all data.

The dropout rate in this case is 0.25 and 0.5, which is a good initial configuration[11].

3.3 Deeper network with Dropout layers

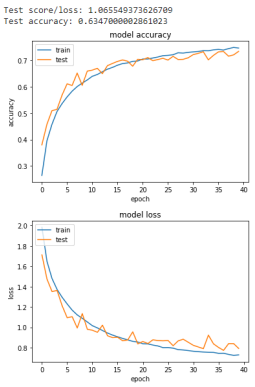
Adding layers in CNN helps more features to be extracted, increasing the weight in the network and in consequence, increasing the model complexity, as shown in Figure 9a, and the chance of overfitting.



Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 32, 32, 32)	896
activation_6 (Activation)	(None, 32, 32, 32)	0
max_pooling2d_4 (MaxPooling 2D)	(None, 16, 16, 32)	0
dropout_2 (Dropout)	(None, 16, 16, 32)	0
flatten_3 (Flatten)	(None, 8192)	0
dense_6 (Dense)	(None, 512)	4194816
activation_7 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 10)	5130
activation_8 (Activation)	(None, 10)	0

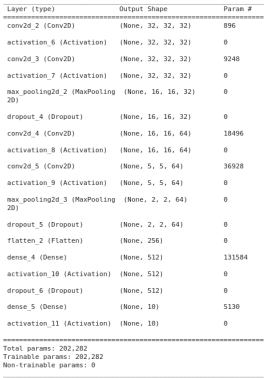
Total params: 4,200,842
Trainable params: 4,200,842
Non-trainable params: 0

(a) Model 2 summary



(b) Result after 40 epoch of model shown in Figure 8a

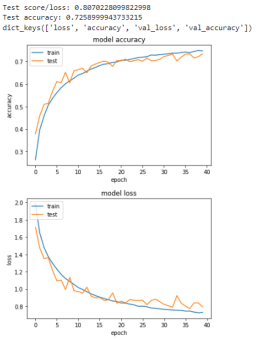
Figure 8: Model 2 and result



Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 32, 32, 32)	896
activation_6 (Activation)	(None, 32, 32, 32)	0
conv2d_3 (Conv2D)	(None, 32, 32, 32)	9248
activation_7 (Activation)	(None, 32, 32, 32)	0
max_pooling2d_2 (MaxPooling 2D)	(None, 16, 16, 32)	0
dropout_4 (Dropout)	(None, 16, 16, 32)	0
conv2d_4 (Conv2D)	(None, 16, 16, 64)	18496
activation_8 (Activation)	(None, 16, 16, 64)	0
conv2d_5 (Conv2D)	(None, 5, 5, 64)	36928
activation_9 (Activation)	(None, 5, 5, 64)	0
max_pooling2d_3 (MaxPooling 2D)	(None, 2, 2, 64)	0
dropout_5 (Dropout)	(None, 2, 2, 64)	0
flatten_2 (Flatten)	(None, 256)	0
dense_4 (Dense)	(None, 512)	131584
activation_10 (Activation)	(None, 512)	0
dropout_6 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 10)	5130
activation_11 (Activation)	(None, 10)	0

Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0

(a) Model 3 summary



(b) Result after 40 epoch of model shown in Figure 9a

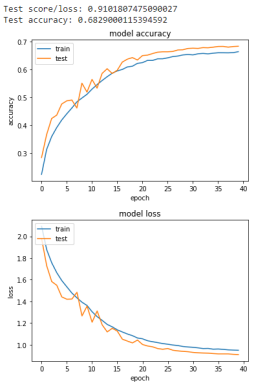
Figure 9: Model 3 and result

However, the accuracy increased almost 10% from model 2 to model 3, 63.47% to 72.59%, suggesting the number of layers added was a good trade-off as the model did not overfit as much. Other than the accuracy, the graph shown in Figure 9b is similar to Figure 8b.

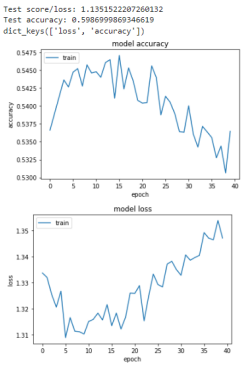
3.4 Reduce Fluctuation

From Figure 9b we can see the accuracy curve does not converge very well and there is fluctuation. Several reasons could cause this:

Small dataset compare to the network: As shown in Figure 9a, there are over 200k trainable parameters and 10k training data. The dataset is small in comparison to the network.



(a) Model summary as shown in Figure 9a with updated learning rate after each epoch



(b) Result after 40 epoch of model 3 shown in Figure 9a with augmented data

Figure 10: Result for learning rate decay and data augmentation

Small batch_size: A small batch_size means outliers would have a bigger effect on the overall result[12]. A batch_size of 128 has been used so far which means only 128 samples are used to make one update to the model parameters, which is a very small size compared to the 10k training data.

Other than the reasons above, which would affect the training time of the model, another possible solution is to update the learning rate after each epoch. A method from Tensorflow.Kera called callbacks.LearningRateScheduler was used as the callback function. The resulting graph is shown in Figure 10a. The accuracy has not increased but it is clear that the testing accuracy curve is much smoother and has a higher accuracy rate than training.

3.5 Data Augmentation

One way to improve the performance of the model is to provide more data for training. Only the training accuracy is presented in Figure 10b as the time consumption is too much and in comparison is not an effective way to improve performance in this case as time is short but training data is not lacking.

Model	Accuracy
Initial Model	65.76
Initial Model + Dropout	63.47
Above + Deeper layers	72.59
Above + Learning Rate callback	68.29

Table 1: Table of all testing accuracy from different CNN models

4 Analysis and Comparison

4.1 Related to dataset

The dataset used for this report is a smaller version of CIFAR10 where 1000 images from each class were used for training instead of 5000, and 200 images from each class were used for testing instead of 1000. This change was made with the time taken for model training in mind, as mentioned in section 2 and 3, the models' uses were time-consuming. However, this severally affected the performance of the models, as Pedro Domingos said, more data bears a cleverer algorithm[10]. However as this is a uniform factor for all experiments conducted for this paper, the effect of parameters can still be assessed and compared albeit the accuracy rate would be low.

4.2 Related to results

As presented in the confusion matrix in Figure 5, BOVW had a hard time predicting the class of the testing data, an overwhelming amount of testing data was classed into two classes. Increasing the number of visual words does not improve this as also shown in Figure 6. This could be because due to the nature of the feature detectors and object in the images, some detectors are more compatible with certain objects, which is why the vocabulary histogram is different as shown in Figure 3a and 3b, because different in features detected bring different in visual words and therefore different in intensity. Using KNN bring a similar problem as shown in Figure 5c. However, from Figure 4 we can see the accuracy generally improves with the number of visual words.

For CNN, it is clear from all but the initial model that more epoch generally leads to better performance. While the model improves, the network topology gets deeper instead of wider. This was by design as wider NN trains with all possible in-

put values but is not as good at generalization as a deeper topology, especially when training with all possible input values is unrealistic due to the time and resource constraints. A deeper network topology, like CNN used here, can capture the hierarchy in the features and the ability to learn different levels of abstraction of the features are the reason they perform well for datasets like this. For all results from the CNN model, the accuracy of the training dataset always improved steadily as the epoch increased. This shows none of the models suffers from underfitting. As shown in Table 1, deeper CNN networks give better accuracy, however, learning rate decay shown in Figure 10a gives the best progression of accuracy.

5 Conclusion

I believe CNN was a more suitable model for CIFAR10. This dataset has images of 10 classes from planes to cars to animals, the variation of the object meant the distinct features from each classes are very different. Going back to Figure 2, in BOVW different feature detector needs to be combined to ensure all possible feature will be detected. This brings more computation requirements to the model.

CNN on the other hand uses layers of filter to detect features and their hierarchy, meaning it is not constrained by one or multiple feature detectors, making it more suitable for CIFAR10.

CNN has constantly outperformed BOVW in these experiments. This could be due to many factors, such as training data size. CNN is harder to comprehend than some of the mathematics involved in BOVW, but given the complete and extended packages available for building CNN networks, it has a lower coding requirement for users and comparatively better performance. To improve the accuracy more, in future I would consider using ResNet instead of building a CNN from scratch as it has been proven to be efficient in classification.

Albeit the performance of BOVW in this experiment, the dataset is admittedly not ideal for a single feature detector to process and there are many parameters involved that can affect the performance as mentioned in section 2.2.1.

References

[1] Tinne Tuytelaars and Krystian Mikolajczyk. *Local invariant feature detectors: a survey*. Now Publishers Inc, 2008.

[2] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[3] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[4] Daliyah S Aljutaili, Redna A Almutlaq, Suha A Alharbi, and Dina M Ibrahim. A speeded up robust scale-invariant feature transform currency recognition algorithm. *International Journal of Computer and Information Engineering*, 12(6):365–370, 2018.

[5] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image matching using sift, surf, brief and orb: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*, 2017.

[6] Pedram Azad, Tamim Asfour, and Rüdiger Dillmann. Combining harris interest points and the sift descriptor for fast scale-invariant object recognition. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4275–4280, 2009.

[7] Lei Chen. *Curse of Dimensionality*, pages 545–546. Springer US, Boston, MA, 2009.

[8] Lindsay I Smith. A tutorial on principal components analysis. 2002.

[9] Dhivya Bharkavi and Grasha Jacob. An analysis of image enhancement based on histogram equalization methods. 5:305–309, 03 2018.

[10] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55:78–87, 10 2012.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[12] Khurram Hameed. Why is my training loss fluctuating?, 05 2021.